М.; СПб., 2013. С. 10–16. URL: http://www.aarheo.ru/components/com_jshopping/files/demo_products/_-_._._21.pdf (20.02.2017); 2) К оценке профессиональной подготовленности таможенных служителей XVII века // Там же. Вып. 6. М.; СПб., 2014. С. 3–7. URL: http://www.aarheo.ru/ components/ com_jshopping/files/ demo_products/Vestnik_A-A-6.pdf (20.02.2017); 3) Археография между Сциллой и Харибдой // Там же. Вып. 7. М.; СПб., 2014. С. 42–54. URL: http://www.aarheo.ru/components/com_jshopping/ files/demo_products/Vestnik_A-A-7.pdf (20.02.2017).

[10] Таможенные книги Сухоно-Двинского пути XVII в. Вып. 3. СПб., 2015; Вып. 4. СПб., 2016.; Вып. 5. СПб., 2017. В печати; Вып. 6. СПб., 2017. В печати.

[11] РГАДА. Ф. 137. Оп. 1. Вязьма. № 31, 33; Устюг, № 80-а.

[12] Там же. Вязьма. № 33. Л. 4.

[13] Там же. Л. 53 об.

[14] *Мерзон А. Ц.* Устюжские таможенные книги XVII в. С. 90–91; РГАДА. Ф. 137. Оп. 1. Устюг. № 80-а.

[15] РГАДА. Ф. 137. Оп. 1. Устюг. № 82. Л. 108, ненум. л. между л. 111 и 112, 158, 176.

[16] А. Ц. Мерзон предполагал иную последовательность создания этих рукописей: с направлявшегося в Москву беловика делалась копия для местной таможенной избы (*Мерзон А. Ц.* Устюжские таможенные книги XVII в. С. 90). Такое мнение вряд ли справедливо из-за указанных выше трудностей при изготовлении белового экземпляра непосредственно с таможенной книги первого вида.

[17] *Мерзон А. Ц.* Устюжские таможенные книги XVII в. С. 89.

[18] Таможенные книги Сухоно-Двинского пути XVII в. Вып. 4. С. 30–31.

[19] В полном виде или во фрагментах известны несколько таких книг: устюжские 1646/47 и 1672/73 гг., Московской большой таможни 1693/94 г., несколько тихвинских и холмогорской таможенной избы 1658 г. (*Базилевич К. В.* К вопросу об изучении таможенных книг XVII в. С. 78; *Сакович С. И.* Из истории торговли и промышленности России конца XVII века. М., 1956; Книги Московской большой таможни — 1693–1694 гг.: Новгородская, Астраханская, Малороссийская. М., 1961; *Тимошина Л. А.* Холмогорская таможенная книга 1658 г. // ОФР. Вып. 4. М., 2000. С. 186–236).

[20] Ср.: *Базилевич К. В.* К вопросу об изучении таможенных книг XVII в. С. 78.

И. Р. Соколовский

ОБРАБОТКА МАШИНОЧИТАЕМЫХ КОПИЙ ТАМОЖЕННЫХ КНИГ XVII в. КАК БАЗ СЛАБОСТРУКТУРИРОВАННЫХ ДАННЫХ С ПОМОЩЬЮ СКРИПТОВ НА ЯЗЫКЕ PYTHON ВЕРСИИ 3.X: НЕКОТОРЫЕ РЕЗУЛЬТАТЫ

Мы хотели бы определить понятие «машиночитаемой копии документа» следующим образом. Машиночитаемой копией документа является такая электронная копия документа XVII в., в которой каждому знаку, присутствующему в документе, сопоставлен соответствующий знак электронной копии, который представлен кодом какого-либо элемента из соответствующей кодовой таблицы. Наше определение машиночитаемости имеет «узкий» харак-

тер, так как предполагается, что документ не только может быть прочитан, но и что из него может быть извлечена определенная информация, то есть цифровая фотография документа не попадает под наше определение «машиночитаемости», но попадает под более широкое определение, так как бинарный файл может быть преобразован в изображение на мониторе или принтере. Таким образом, электронная копия документа хранится в электронном виде (то есть в виде намагниченности, наэлектризованности, полупроводника с измененным состоянием или как-то еще с использованием электрического тока) и текст документа может быть представлен набором «бинарных» символов, которые машина преобразует для нас в буквы кириллического алфавита и арабские (или римские) цифры, выводя их на какое-то графическое или печатное устройство. С этими буквами и цифрами мы можем оперировать точно так же, как мы проделываем это с буквами и цифрами напечатанными на бумаге или написанными от руки. Однако тут существует одно важное отличие. Строчные и прописные буквы в машинном представлении текста закодированы разными элементами кодовой таблицы («кодовой страницы» на жаргоне программистов).

Подобное различение позволяет нам использовать прием поиска в электронной копии текста, который носит название «поиска с использованием регулярных выражений». «Регулярными выражениями» называется мета-язык, описывающий группы символов обычного языка таким образом, чтобы машина могла обнаружить их в заданном тексте и сохранить в месте, указанном пользователем. Примером регулярного выражения может служить такое описание «два слова с прописной буквы, идущие подряд и не разделенные знаками препинания, а только пробелом» (1). Или, например, такое «четыре цифры, идущие подряд» (2). Или такое, «группа символов, начинающаяся с заглавной буквы и заканчивающаяся точкой» (3). С помощью регулярного выражения (1) мы можем обнаружить, например, все имена и фамилии, встречающиеся в тексте. Хотя, надо отметить, что в нашу выборку попадет и часть географических названий, например, «Западная Сибирь» или «Северная Евразия». С помощью выражения (2) в поле нашего внимания попадут все даты после 1000 г. («года») встречающиеся в тексте, но так же и любые четырехзначные числа, например, «2359» или «4456». Чтобы исключить их мы должны уточнить наш поиск, наложив дополнительные ограничения и усложнив наше регулярное выражение так, чтобы оно исключило все числа, находящиеся на числовой оси после числа «2017». Наконец, регулярное выражение (3) теоретически должно было бы дать нам все повествовательные предложения, встречающиеся в тексте, но оно будет корректно работать только для художественных текстов. В научных и научно-популярных текстах из-за обилия разных сокращений («г.», «в.» и т. д.) мы получим множество разрозненных и не очень информативных фрагментов. Однако замена в регулярном выражении точки на вопросительный или восклицательный знак вернет нам все вопросы или восклицания, что для научного текста может оказаться очень интересным экспериментом. И уж тем более, это может быть интересным для группы текстов, разнесенных по времени. Если же мы обнаружим подобные символы в тексте таможенной книги, то это автоматически будет сигнализировать об ошибке набора.

Язык регулярных выражений позволяет частично или полностью автоматизировать обнаружение в тексте таких структур, которые мы обнаруживаем при обычном чтении. Например, мы различаем в тексте таможенной книги преамбулу и тексты отдельных «явок», можем отличить зафиксированные сделки с рыбой от сделок с мукой или сделок по обмену лошадьми и так далее. Прочитанные для нас фрагменты текста программа может сохранить в удобное для нас место, а счетчик может указать, сколько именно фрагментов текста соответствуют нашим критериям.

Фактически операция использования «регулярных выражений» и иных «шаблонов» оказывается эквивалентной операции разнесения фрагментов таможенной книги по ячейкам электронной таблицы для последующего обсчета, предшествующего написанию исследовательского текста.

Такая операция возможна, если мы воспользуемся концепцией слабоструктурированных данных. Эта концепция предполагает, что данные таможенной книги имеют определенную структуру, которая отличает их от данных, которые, например, содержатся в художественных произведениях или дневниках путешественников.

В отличие от художественных произведений, таможенные книги разбиты на фрагменты с повторяющейся структурой. Например, в них присутствуют какие-то данные об обмене лошадьми или совершении других коммерческих операций. Например, «Явил продать луку саженцу по оценке на 10 р. Там. кн. II, 444. 1652 г.» [1]. Эти фрагменты текста носят формульный характер. Они начинаются с определенных языковых клише и заканчиваются таковыми. Переменными в этих фрагментах являются данные о составе продавцов, покупателей, количественные и качественные характеристики обмененных товаров. Эта особенность приказного делопроизводства XVII в. позволяет нам выделить из текста данные о торговых операциях над отдельными товарами («русский товар», рыба, лошади) не только по содержанию, но и по форме таможенной записи.

Таким образом, текст таможенной книги является определенным образом структурированным. При передаче его для научно-критической публикации по правилам 1990 г. в текст источника вносятся дополнительные структурирующие элементы: знаки препинания, прописные буквы, передача чисел арабскими цифрами вместо кириллических букв с дополнительными знаками.

Однако такая слабоструктурированная информация отличается от жестко структурированной информации обычной базы данных, в которой данные разнесены по столбцам и строкам, чтобы в каждой ячейке оказались данные определенного типа и даже, как это часто бывает, определенной длины.

Такое жесткое представление данных облегчает их поиск и сортировку, однако очень часто оказывается неприемлемым для исторического исследования, поскольку историка очень часто интересует не только типическое и повторяющееся, но и уникальное, неповторимое, которое при таком представлении данных оказывается исключенным.

Если мы уже располагаем машиночитаемой копией исторического документа XVII в., которая содержит некоторую слабоструктурированную инфор-

мацию, то каковы те машинные средства, которыми мы могли бы обработать эту электронную копию для получения нужных нам сведений?

Проблемы исследователя с так называемой «гуманитарной подготовкой» в этой ситуации оказываются проблемами двоякого рода. Во-первых, это дилемма пострашнее, чем у Буриданова осла: есть десятки, если не сотни языков программирования, доступных современным пользователям операционных систем, (например, Python наследует черты десяти разных языков). Во-вторых, общая подготовка в области овладения компьютерными навыками создает у исследователя-гуманитария фальшивый образ сложности и недоступности программирования для «простых смертных». Надо отметить, что если программирование в некоторых современных областях действительно довольно сложно и требует специальной подготовки и многолетнего опыта, то те области, которые относятся к компетенции историка — довольно просты и на начальном уровне требуют только таких навыков программирования, которые под силу освоить любому школьнику старших классов. Надо принять во внимание еще и то, что навыки и объем знаний, которые обычно требуются от историка, гораздо сложнее и больше, чем те навыки, которые требуются от любого программиста.

Мы разрешили проблему выбора «первого» языка программирования волюнтаристским образом, ориентируясь на советы некоторых студентов Факультета информационных технологий Новосибирского государственного университета (больше всего, Андрея Витальевича Таранцова, выпуск 2007 г.). Предметом нашего выбора стал язык Python (произносится, примерно как «Пайсон», однако в русском языке распространено его русифицированное название «Питон»).

Этот язык характеризуется как высокоуровневый язык программирования общего назначения, ориентированный прежде всего на повышение производительности разработчика и читаемости кода. Поэтому он очень удобен для начинающих, особенно тех, кто склонен допускать синтаксические и орфографические ошибки, которые сразу становятся видны. Интерпретатор прерывает выполнение программы при любой орфографической ошибке, что тоже очень удобно для исправления ошибок и отладки кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объем полезных функций. Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. В данном случае мы имеем как раз дело с тем моментом, в котором не легко разобраться без детального опыта в программировании, так как в этой области программистами сломано немало копий. Надо отметить, что основные архитектурные черты этого языка — динамическая типизация, автоматическое управление памятью, полная интроспекция (это означает, что для любого объекта можно получить всю информацию о его внутренней структуре), механизм обработки исключений (посредством операторов try, except, else, finally, raise), поддержка многопоточных вычислений — оказывают разное влияние на пользователя с историческим образованием. Например, динамическая типизация затрудняет обучение

программированию, потому что историк обычно не различает «1» как текст и как натуральное число и Python не заставляет его этому научится (женщины-историки традиционно более внимательны к таким вещам, чем мужчины-историки). Автоматическое управление памятью приводит к непропорционально щедрому расходованию оперативной памяти, замедляет выполнение программ и требует от программирующего историка специальных усилий в этой области. Поддержка многопоточных вычислений обычно оказывается ненужной.

Код в Python организовывается в функции и классы, которые могут объединяться в модули (они, в свою очередь, могут быть объединены в пакеты), что при некотором навыке позволяет широкое повторное использование кода и экономит много сил и средств, повышая производительность исследователя при работе в качестве квази-программиста.

Важной особенностью Python является то, что он распространяется под свободной лицензией, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные.

Еще важно отметить, что Python портирован и работает почти на всех известных платформах: под Microsoft Windows, практически все варианты UNIX (включая FreeBSD и Linux), Mac OS и Mac OS X, iPhone OS 2.0 и выше, OS/2, Windows Mobile, Symbian и Android.

3 декабря 2008 г., после длительного тестирования, вышла первая версия Python 3.0. В ней устранены многие недостатки архитектуры с максимально возможным (но не полным) сохранением совместимости со старыми версиями Python. Однако обилие библиотек, написанных для Python 2.0, привело к тому, что на сегодня поддерживаются обе ветви развития (Python 3.x и 2.x), что не очень-то удобно для пользователя- гуманитария.

Более подробно с возможностями языка Python можно познакомиться, например, на сайте Викиуниверситета[2]. Надо отметить, что освоение языка Python облегчается фактом его популярности: существует множество книг, online курсов и видеофильмов на различных языках, которые помогают освоить этот язык почти на любом уровне — от базового до продвинутого.

Таким образом, Python является полноценными языком программирования, что позволяет его использовать для обработки данных таможенных книг XVII в.

В заголовке данного текста мы использовали слово «скрипт» (сценарий), чтобы отличать используемые программные продукты от того, что принято еще называть «приложениями», которые обычно у гуманитариев ассоциируются с программированием. Одно из программистских определений сценария гласит, что «сценарии обычно интерпретируются, а не компилируются» [3].

На наш взгляд существует два пути избегания субъективности в подходе к историческому документу. Первый путь — следование тщательно выработанной исследовательской программе, составленной с участием других исследователей. Второй путь — на наш взгляд, как ни парадоксально это звучит, составление частотного словаря. Поскольку частотный словарь является самым простым средством извлечения информации из текста.

Частотный словарь нелемматизированных словоформ позволяет 1) определить фокус внимания текста; 2) выявить ошибки и опечатки; 3) составить исчерпывающий список тем, которые затронуты в данном тексте. Самые часто встречающиеся слова (1) определяют о чем, собственно, повествует данный текст. Было бы странно, например, обнаружить, что самое часто встречающиеся слово в тексте о зерне — «масло». Если мы обнаружим, что у нас есть слова, незначительно различающиеся по составу букв, но обладающие разной частотой, например, одно встречается чаще, а другое один или два раза, то есть смысл проверить, не идет ли речь об опечатке (2). Наконец, список всех нелемматизированных словоформ исчерпывающим образом описывает все темы, которые подняты в данном тексте (3). Обычно при составлении частотных словарей проводится операция лемматизации, то есть приведение всех слов, встречающихся в тексте, к их нормальной форме. Для наших целей такая операция избыточна, мы готовы работать с существительными в любом падеже, роде и числе. То же самое относится и другим частям речи. Однако такая работа может быть проведена филологами, если они будут изучать историю русской лексики по изданным текстам таможенных книг. Кроме частотного словаря, отсортированного в алфавитном порядке (например, вот так: юфтей — 1; юфти — 2; юхтей — 9; юхти — 4; юхть — 3; яблоков — 1) можно еще составить словарь с «примерами», то есть для каждого слова подобрать фрагмент текста определенной длины, где оно встречается хотя бы один раз. Проблема такого словаря заключается в объеме. Из-за многократного повторения фрагментов текста исходный текст объемом 62 332 знака влечет создание текста объемом 4 700 811 знаков, где 14 151 раз встречаются ключевые поля. Обработка такого файла на обычном домашнем персональном компьютере превращается в проблему. Текстовый или html файл объемом 7–8 Мб откроет не каждый редактор и не всякий браузер. Это, правда, самый простой путь обработки текстов, на самом деле под исследовательский проект может быть написано любое программное обеспечение, дело только за финансированием и желанием исследователя.

Как это работает? Приведем пример. Возьмем публикацию таможенной книги Вологды 1634/35 г. [4] Выберем произвольный фрагмент текста источника объемом 62 332 знака, то есть полтора авторских листа. Работа с таким фрагментом текста с занесением всех содержащихся в нем данных в таблицу может занять около месяца рабочего времени. Будем исследовать текст самым простым методом: методом составления словаря нелемматизированных словоформ. Мы написали очень простой скрипт на языке Python, который различает строчные и прописные буквы и в котором точки и тире слепленные со словами превращают слово в отдельную единицу анализа, что иногда оказывается удобным.

Если мы посмотрим частоту встречаемости сокращения «л.», то увидим, что оно встречается 127 раз. То есть в нашем фрагменте 127 листов с оборотами. Поскольку нам хорошо известен формуляр документа, то мы понимаем, что слово «явил/явили» встречается в каждой отдельной явке товара и таких упоминаний в нашем фрагменте 211, в том числе 6 коллективных (л. 176 об.,

196 об., 226, 230 об., 234 об.). Слово «платил», свидетельствующее о реальной уплате пошлины, встречается 168 раз, а слово «взято» — 228 раз. Мы можем легко посмотреть состав торговцев: верховаженинов — 2, вологженинов — 27, костромитинов — 29, тверитинов — 1, тотьменинов — 2, угличан — 4, устюжан — 1, белосельцев — 3, голландцев — 1, даниловцев — 19, дмитровцев — 2, каргопольцев — 5, козенинцев — 5, любимцев — 5, москвичей — 6, пошехонцев — 4, романовцев — 3, ростовецев — 7, Сямские волости — 1, чаронженинов — 22, чухломецев — 2, юрьевчан — 1, ярославцев — 9. Стоит проверить, для всех ли явок у нас есть состав торговцев. Можно, конечно, просуммировать эти данные вручную, но можно и написать соответствующий скрипт.

После применения этого скрипта мы получим результат, что нам известно происхождение 161 чел. и контрольную информацию, массив натуральных чисел в этом тексте, чтобы убедиться, что мы правильно все подсчитали: 2, 27, 29, 1, 2, 4, 1, 3, 1, 19, 2, 5, 5, 5, 6, 4, 3, 7, 1, 22, 2, 1, 9. Собственно, таким образом мы можем решить любую из традиционных исследовательских проблем. Например, нас интересует торговля медью, освещенная в данном фрагменте. Мы смотрим файл частотного словаря. Там обнаруживается, что слово «медь» встречается в тексте нашего фрагмента три раза и все разы в форме «меди». Обращаемся к нашему файлу, где словарь дан с «примерами», то есть всеми фрагментами текста, разделенными точкой, где встречается данное слово. Получаем вот такую выборку:

меди *Пример*: каргополец Федор Федосеев явил с товарыщом с Семеном в проезд на 3 санех 100 кож, 5 подставов сукон полуаглинских, 9 котлов красные меди;

меди *Пример*: взято государевы таможенные пошлины 15 ал. 4 д. костромитин Маныло Осипов явил в проезд 6 подставов сукон аглинских, 4 п. меди в котлех;

меди *Пример*: меди, на 4 р. мелочи.

Это соответствует трем явкам, которые попали в наш фрагмент таможенного текста: явки каргопольца Федора Федосеева, у которого было 9 «котлов красные меди», ярославца Парфения Выморова, у которого было 12 фунтов меди в составе его товаров на сумму 55 руб., костромитина Маныло Осипова, у которого было с собой на продажу 4 пуда «меди в котлех». Из этих данных можно делать уже различные выводы, от которых мы воздержимся, так как взяли для своих исследований не целый текст таможенной книги, а всего лишь произвольный фрагмент ее.

Традиционный способ обработки таможенных книг заключается в разделении операций: 1) подготовка текста к изданию; 2) сбор информации из таможенных книг; 3) обработка этой информации с целью получения картины торговли XVII в. по данным таможенных книг. С началом компьютеризации исторической науки обычно операция (1) осуществляется в любом текстовом редакторе (Word, OpenOffice.Write, ViM и т. д.), а операции (2) и (3) выполняются с помощью электронных таблиц (Excel, OpenOffice.Calc и т. д.). Таким образом, одна и та же информация переписывается два и более раз, что создает большое пространство для ошибок, опечаток, пропусков и повторов. Это мо-

жет привести к существенному искажению информации о состоянии торговли в XVII в. по данным таможенных книг. Использование скриптов на языке Python позволяет нам работать с одной и той же электронной копией текста таможенной книги без ручного переписывания ее данных из одной формы в другую. Это, несомненно, снижает вероятность появления ошибок, вызванных усталостью, невнимательностью, «замыленностью» глаза и т. д., поскольку электронное устройство не устает и не отвлекается. Если же мы допустили какую-то ошибку при написании скрипта, то она будет носить систематический и единообразный характер, и, соответственно, может быть исправлена так же единообразно и быстро, как и появилась.

Недостатком данного метода является довольно трудоемкое создание выборок по разным критериям, что является основной эвристической ценностью баз данных. Если нас, например, интересуют все «лалетины», продававшие рыбу с февраля по март, то легкое создание запроса при табличной форме представления в обычной базе данных превратится в довольно трудное написание скрипта на языке Python или ином другом языке. Поэтому в настоящее время это оправдано в рамках коллективной работы или в рамках написания большого исследования в виде монографии или цикла статей.

Следует, однако, отметить, что базы данных, создаваемые на основе одной или нескольких таможенных книг, не являются полноценными многопользовательскими базами данных. Это исследовательские базы данных, существование которых обусловлено временными рамками реализации той или иной исследовательской программы.

Если нас интересует реальная база данных всех записей таможенных книг, то необходимо переводить эти записи в формат какой-либо жестко структурированной базы данных, поддерживающей хранение данных в табличном виде.

Если нам нужна база данных, к которой мы обращались бы многократно, то одним из вариантов было бы написание скрипта, который переводил данные из текстового вида в формат csv — comma separated values, то есть сущностей, разделенных запятыми. Это текстовый формат, который не хранит ничего кроме символов из существующих кодовых таблиц (скажем, ANSCII или Unicode) с разделителями «запятая», «точка» или «кавычка», позволяющими соответствующим программам легко преобразовать его в электронную таблицу, легко импортируемую любой базой данных.

Отдельной темой, в которую мы не вступаемся в данной статье, является тема создания корпусов русского языка XVII в. на основании дипломатических наборных публикаций таможенных книг. Причина нашего избегания этой темы кроется не только в том, что мы не владеем достаточно сложной методикой создания корпусов, но и тем, что в нашей практике не попадалось дипломатических изданий таможенных книг, а только научно-критические, осуществленные по правилам издания исторических документов 1990 г.

Примечания

[1] Словарь русского языка XI–XVII вв. Вып. 23. М., 1996.

[2] URL: https://ru.wikiversity.org/wiki/Примеры_программ_на_языке_Python (20.02.2017).

[3] *Ousterhout J.* Scripting: higher-level programming for the 21st Century // IEEE Computer. 1998. Vol. 31, N 3. P. 23–30. URL: http://web.stanford.edu/~ouster/cgi-bin/papers/scripting.pdf (20.02.2017).

[4] Таможенная книга города Вологды 1634–1635 гг. / Сост. Е. Б. Французова. М., 1983.